Master's Thesis

Improved Prediction of Deep Temporal Neural Networks with Trend Filtering

Youngjin Park

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

2021

Improved Prediction of Deep Temporal Neural Networks with Trend Filtering

Youngjin Park

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

Improved Prediction of Deep Temporal Neural Networks with Trend Filtering

A thesis/dissertation submitted to
Ulsan National Institute of Science and Technology
in partial fulfillment of the
requirements for the degree of
Master of Science

Young-Jin Park

11. 23. 2020

Approved by

Advisor
Kwang-In Kim

Improved Prediction of Deep Temporal Neural Networks with Trend Filtering

Young-Jin Park

This certifies that the thesis/dissertation of Young-Jin Park is approved.

11. 23. 2020

Signature

Advisor: Kwang-In Kim

Signature

Jaesik Choi: Thesis Committee Member #1

Signature

Byoung-Ki Seo: Thesis Committee Member #2

Abstract

Forecasting with multivariate time series that has been studied for a long time has goal to predict future values using previous and current values. Since it is difficult to know how many noise is intrinsic within fast changing time series data, training the model having high predictive performance is difficult. Recently, many researchers have been interested in many deep neural network such as recurrent neural networks, encoder-decoder structure, and neural networks with attention mechanism to design them to forecast future financial time series data. Many attempts have been to use those methods for highlighting more important features for capturing the long period of temporal dependencies in the task of forecasting multivariate time series data to make predictive performance better. In this paper, a new framework is introduced, which is utilized deep temporal neural networks with trend filter techniques that make original time sequence data mixed with the noise into the trend. We show that the performance of each deep neural network using a feature temporally processed by trend filter skills. To verify this proposed method, three state-of-the-art models for forecasting future values in finance stock market sequence, are used for comparing the predictive performance with the original method. Expanded experiments for conducting the t-test between this proposed method and original one show that the new framework is useful and has statistical significance of the fact that proposed method achieves better performance.

Contents

List of Figures

List of Tables

I Introduction

Designing predictive algorithms for multivariate time series have been used in various field, such as forecasting financial market [1], weather forecasting [2], groundwater level prediction [3], and traffic predictions [4]. Also, a few methods have been tried for forecasting time series such as ARIMA [5–7], Gaussian Processes [8–11]. Even with these efforts, it is still challenging to predict these data having complicated and non-linear dependencies not only between time but also among several time series features, which can fluctuate dynamically at each time step.

The recurrent neural network (RNN) [12–14] is a kind of deep neural network for sequence forecasting with shared internal parameters. This network is vigorously used for forecasting tasks in time series to capture nonlinear patterns. However, the classic RNNs are related with not only the vanishing or forgetting gradient problems but also having difficulty when used to detect long-term representation. Recently, long short-term memory (LSTM) [15–17] and the gated recurrent unit (GRU) [18, 19] had some success in solving these limitations and have used in diverse areas, *e.g.*, neural machine translation [20, 21], and speech recognition [22]. Based on encoder-decoder structures [18, 23, 24] also helps to capture not only the input features but also long temporal patterns by selecting the proper parts among all encoder hidden states.

Dual-stage Attention-based RNN (DA-RNN) [25] is well devised with encoder-decoder structure having a two-stage attention [26–29] structure both to focus appropriate input features and to take the temporal sequential pattern of the input data. A DA-RNN uses exogenous information that the last time time step T of whole input features is included in the target time steps that the model is to predict. On the other hand, Dual Self-Attention Network [30] forecasts future values without exogenous information unlike DA-RNN and feeds each of the multivariate time series data individually into two parallel convolutional components. In addition, those representations is fed into self-attention modules which is from transformer network [31, 32]. All experiments are conducted without exogenous information for all deep temporal neural networks in this paper because exogenous information contains the future values of input features.

It is not simple task to figure out the extent how many the noise from the important signals in time series finance data. In addition, structural break predictions are necessary to estimate future values. However in noisy financial times series, observing structural breaks is challenging. This complex structure of time series data interrupt appreciate predictions and make the predictive performance of deep neural networks to be worse. To solve these problems and to increase the performances capabilities for prediction problem, we propose a new framework that adds a feature using the trend filtering technique such as Low-pass filter (LPF) [33, 34], L1 Trend Filtering (L1TF) [35] to the input values.

L1 trend filtering captures a piecewise pattern between two points of trend changes and uses a hyperparameter λ [36]. Since λ, which is capable of controlling the smoothness among the trend and the size of residual for the trend and actual data, is optimally decided by a lot of experiments, it can solve the problem that are difficult to distinguish between information signals

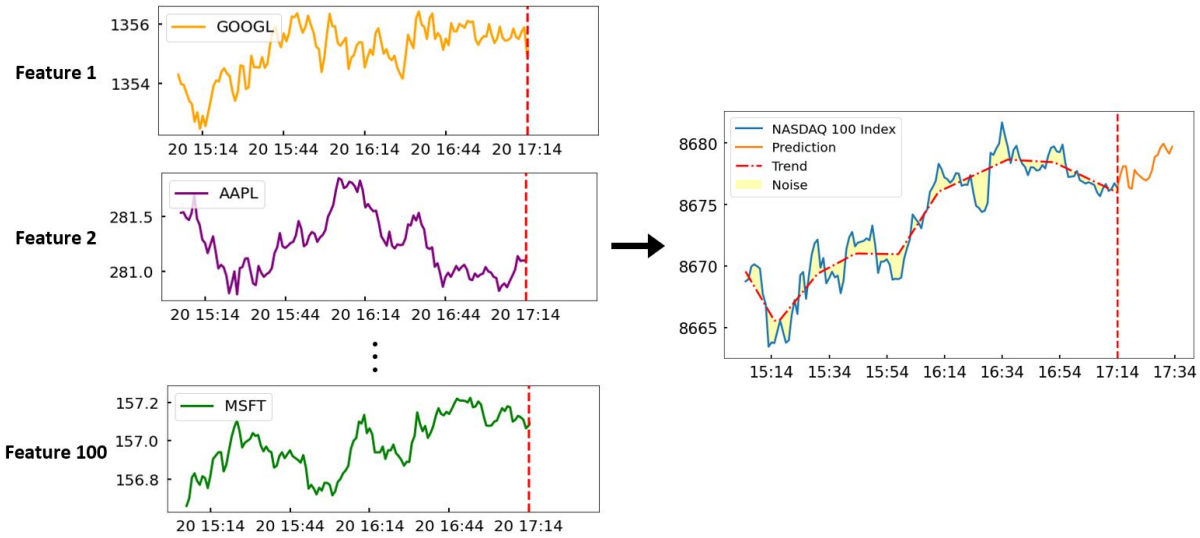

Figure 1: Forecasting with multivariate time series, which is NASDAQ 100 Stocks. 100 features is the stock price listed on NASDAQ 100 Stocks, and blue line is about NASDAQ 100 Index. Using those previous and current several times series data, the future values of NASDAQ 100 Index will be predicted. Since it is difficult to estimate the degree of noise mixing on signal information within finance time series, the method for training a well-predictive model is difficult.

and noise so that the model can make proper predictions. On the other hands, low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency hyperparameter. Also selected cutoff frequency hyperparameter is optimally determined by a lot of experiments, the problem that has difficulty in extract noise within noisy time series to make the predictive performance of the model. To verify the effectiveness of this method, we use two cases, one is that trend filter is used and the other is that filtering is not applied, in conjuction with a set of stock index data with many driving series, *e.g.*, the NASDAQ 100 Index, the EURO STOXX 60 Index, the Dow Jones Industrial Average, the FTSE 100 Index, and the TSX 60 Index. Extensive experiments with several time series, which means multivariate time series, demonstrate that the this proposed method is useful and surpasses the baseline method in paired t-test [37] between models with and without the trend filter feature.

Our main contribution in this thesis are as follows:

- A new framework is proposed in this paper, which utilizes deep temporal neural networks with a trend filtering feature, which is a trend of the target value.
- We demonstrate that the performance of deep neural networks using a temporally processed by trend filter skills.
- To evaluate this method, three deep temporal neural networks, which is state-of-the-art in forecasting future time series data, are utilized for comparing between the models with trend filtering feature as an input data and the models without one.

- Extensive experiments on real-world multivariate time series data show that the this proposed method is effective and significantly better than existing original methods.

II Background

2.1 Autoregressive Integrated Moving Average (ARIMA)

By integrating an autoregressive (AR) [38, 39] with a moving average model (MA) [40, 41], the future values are assumed to be a linear function of multiple past observations and random errors. Thus the basic process that generates the time series has the following form:

$$\begin{aligned} y_t = \theta_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} \\ + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}, \quad d = 0, \end{aligned} \tag{1}$$

where y_t and ε_t are the actual value and random error, respectively, at time t. ϕ_i $(i = 1, 2, ..., p)$ and θ_j $(j = 0, 1, 2, ..., q)$ are the parameters of AR and MA estimated using an optimization procedure that minimizes the sum of square errors or some other appropriate loss function. p and q are integers that pertain to the order of the model. d refers to differencing, which is used to calculate the difference between subsequent observations to indicate the stationarity of the time series. One of the important parts of the ARIMA model is that for making the proper model, we set d=0 among (p, d, q) hyperparameters in this paper.

2.2 Deep Neural Network

DNN [42, 43] is a non-linear method that approximates the function F that maps the input $\mathbf{X}$ to the output $\mathbf{y}$ as follows,

$$\mathbf{y} \approx F(X), \tag{2}$$

where F represents the entire layer of DNN. The several equation from the first layer to the output layer can be represented as follows,

$$\begin{aligned} \mathbf{f}_1(\mathbf{X}) &= g_1(W_1\mathbf{X} + \mathbf{b}_1), \\ \mathbf{f}_2(\mathbf{X}) &= g_2(W_2\mathbf{f}_1(\mathbf{X}) + \mathbf{b}_2), \\ \mathbf{f}_i(\mathbf{X}) &= g_i(W_i\mathbf{f}_{i-1}(\mathbf{X}) + \mathbf{b}_i), \\ \mathbf{f}_{L-1}(\mathbf{X}) &= g_{L-1}(W_{L-1}\mathbf{f}_{L-2}(\mathbf{X}) + \mathbf{b}_{L-1}), \\ \mathbf{y} &= \mathbf{f}_{L-1}(\mathbf{X})^T W_L + \mathbf{b}_L, \end{aligned} \tag{3}$$

where $\mathbf{W_i}$ and $\mathbf{b_i}$ represent correspondingly the weight and bias for the i-th layer map, $\mathbf{f_i}$, while g_i is an linear activation function for layer i. Generally, for each layer of a DNN, non-linear mapping is applied to the input with activation functions such as a tanh, ReLU, logistic, sigmoid and softmax.

2.3 Fully Convolutional Network (FCN)

A fully convolutional network (FCN) [44, 45] have been used in image semantic segmentation, which is taking the input image and producing the output with useful inference and learning.

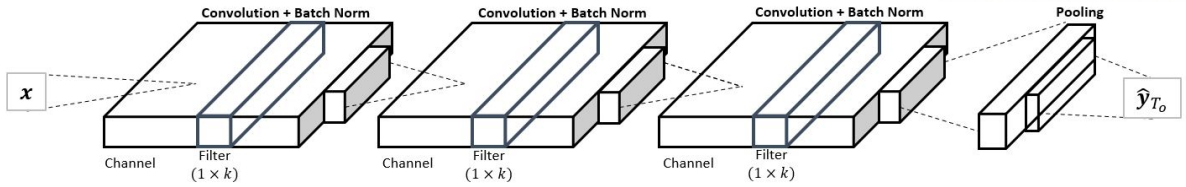

Figure 2: The architecture of Fully Convolutional Network (FCN).

Each layer of data in a convolutional network has three dimensions, $h \times w \times d$, where h and w denotes each dimension of height and width in image data and d denotes the number of features. Receptive fields is mapping from the area in high layer to the area in next layer. These basic components (convolution, batch normalization, pooling, and activation functions) are computed on local input regions.

However, for time series data, some basic component of FCN must be changed to receive multivariate time series as input functions. As it is not spatial, but temporal, the kernel size of one layer is $1 \times k$, where k is the number of input time steps (window size) of one layer. $\mathbf{x}_t$ represents a vector of all input features at time t in a layer, and $\mathbf{y}_t$ is a vector for the following layer. This model is defined as follows,

$$\mathbf{y}_t = f_{ks}(\{\mathbf{x}_{st+\delta}\}_{0 \leq \delta \leq k}), \tag{4}$$

where s is the stride, k is the filter size or kernel size, and the layer type(i.e., convolution structure, a temporal max for max pooling or average pooling) is determined by f_{ks}.

This equation is maintained with filter size and stride length following the below rule,

$$f_{ks} \circ g_{k's'} = (f \circ g)_{k'+(k-1)s',ss'}. \tag{5}$$

An FCN naturally computes a non-linear filter, which is taking the input image and producing the output.

2.4 Dual-stage Attention-Based Recurrent Neural Network (DA-RNN)

A DA-RNN has two attention mechanisms, which is an encoder with input attention to select relevant driving series appropriately at each time step and a decoder with temporal attention to capture related encoder representations during whole time steps. According to such mechanisms, the DA-RNN can capture relevant input features and the long-range temporal information of a time series adaptively. This attention method can predict future target values effectively in multivariate time series finance data.

Given $\mathbf{n}$ driving series, $\mathbf{X}$ denotes input multivariate time series *i.e.* $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^n)^\top = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T_i}) \in \mathbb{R}^{n \times T_i}$, where T_i is the input window size and n is the number of incoming series, which means the number of input features. Typically, the input index time series is represented as $(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{T_i})$. The goal of DA-RNN model is mapping of the future sequence of the target

$\mathbf{y}_{T_o}$, where T_o is defined as the output time step size:

$$\hat{\mathbf{y}}_{T_o} \approx F(\mathbf{y}_1, ..., \mathbf{y}_{T_i}, \mathbf{x}_1, ..., \mathbf{x}_{T_i}), \quad T_o > T_i, \tag{6}$$

where $F(\centerdot)$ denotes a non-linear function for mapping of the future values of the target.

The sequence to sequence architecture is basically the RNN structures that encode each input time series to each representations and decode those representations to output the results. The DA-RNN model uses the LSTM structure as the encoder and decoder to capture the temporal dependent information. Each LSTM layer of the encoder and decoder has a memory cell that has input hidden states, such as $\mathbf{s}_t$ at time t and hidden state $\mathbf{h}_t$ as the output. In the memory cell, the three sigmoid gates is controlled, which are the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$, and the output gate $\mathbf{o}_t$. The LSTM layer can be represented as follows,

$$\begin{aligned}
\mathbf{f}_t &= \sigma(W_f[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(W_i[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_i), \\
\mathbf{o}_t &= \sigma(W_o[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_o), \\
\mathbf{s}_t &= \mathbf{f}_t \odot \mathbf{s}_{t-1} + \mathbf{i}_t \odot \tanh\left(W_s[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_s\right), \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{s}_t\right),
\end{aligned} \tag{7}$$

where $[\mathbf{h}_{t-1}; \mathbf{x}_t] \in \mathbb{R}^{m+n}$ is a combination of $\mathbf{h}_{t-1} \in \mathbb{R}^m$ and $\mathbf{x}_t \in \mathbb{R}^n$, which means the previous hidden state and the current input data. $W_f, W_i, W_o, W_s \in \mathbb{R}^{m \times (m+n)}$, and $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_s \in \mathbb{R}^m$ are trainable parameters.

Continuously, an input attention mechanism in DA-RNN can refer to the previous hidden state $\mathbf{h}_{t-1}$ and cell state $\mathbf{s}_{t-1} \in \mathbb{R}^m$ in the encoder LSTM layer as follows,

$$e_t^k = \mathbf{v}_e^\top \tanh\left(W_e[\mathbf{h}_{t-1}; \mathbf{s}_{t-1}] + U_e \mathbf{x}^k\right), \tag{8}$$

and

$$\alpha_t^k = \frac{exp(e_t^k)}{\sum_{i=1}^{n} exp(e_t^i)}, \tag{9}$$

where T is the input time step, $\mathbf{x}^k \in \mathbb{R}^T, \mathbf{v}_e \in \mathbb{R}^T, W_e \in \mathbb{R}^{T \times 2m}$, and $U_e \in \mathbb{R}^{T \times T}$ denotes the parameters to train.

Using the previous decoder hidden state $\mathbf{d}_{t-1} \in \mathbb{R}^p$ and cell state $\mathbf{s}'_{t-1} \in \mathbb{R}^p$, temporal attention with the decoder can calculate the attention weight of each encoder hidden state at time t as follows,

$$l_t^i = \mathbf{v}_d^\top \tanh\left(W_d[\mathbf{d}_{t-1}; \mathbf{s}'_{t-1}] + U_d \mathbf{h}_i\right), \quad 1 < i \leq T, \tag{10}$$

and

$$\beta_t^i = \frac{exp(l_t^i)}{\sum_{j=1}^{T} exp(l_t^j)}, \tag{11}$$

where $[\mathbf{d}_{t-1}; \mathbf{s}'_{t-1}] \in \mathbb{R}^{2p}$ is the combination between $\mathbf{d}_{t-1}$ and $\mathbf{s}'_{t-1}$, which denotes the previous hidden state and the cell state from the decoder layer, $\mathbf{v}_d \in \mathbb{R}^m, W_d \in \mathbb{R}^{m \times 2p}$, and $U_d \in \mathbb{R}^{m \times m}$.

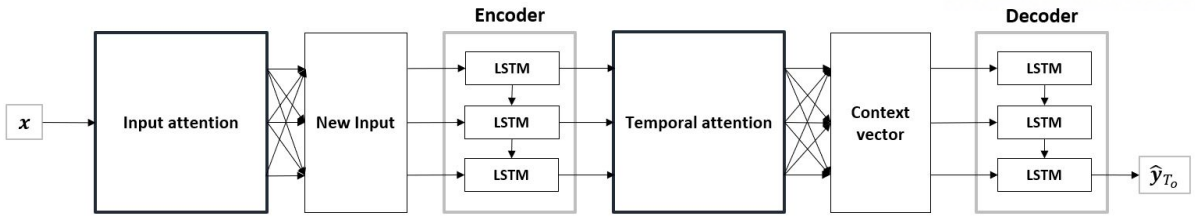

Figure 3: The architecture of Dual-stage Attention-Based Recurrent Neural Network (DA-RNN).

The attention mechanism of the DA-RNN can compute the context vector $\mathbf{c}_t$, which is a weighted sum between temporal attention weight and encoder hidden states $\mathbf{h}_1, ..., \mathbf{h}_T$,

$$\mathbf{c}_t = \sum_{i=1}^{T} \beta_t^i \mathbf{h}_i, \tag{12}$$

where the context vector c_t can be represented at individual time step. Therefore, $\hat{y}_{T_o}$ is calculated as follows,

$$\begin{aligned} \hat{y}_{T_o} &= F(\mathbf{y}_1, ..., \mathbf{y}_T, \mathbf{x}_1, ..., \mathbf{x}_T) \\ &= \mathbf{v}_y^\top (W_y \mathbf{d}_T + \mathbf{b}_w) + b_v, \end{aligned} \tag{13}$$

where the parameters $W_y \in \mathbb{R}^{p \times p}$ and $\mathbf{b}_w \in \mathbb{R}^p$ are mapping to the size of the decoder hidden states, and $\mathbf{v}_y \in \mathbb{R}^p$ and bias $b_v \in \mathbb{R}$ are the weights of a linear function, leading to the final prediction output. This part differs from the concepts in the DA-RNN paper. We did not use the context vector to create the prediction output due to the increased performance.

2.5 Dual Self-Attention Network (DSANet)

DSANet can be used to be more accurate and robust prediction problem for multivariate time series. In this paper, Same problem statement is also used in Eqn. 6. Two parallel convolutional components feeding each of the univariate time series individually is used, which is called global temporal convolution and local temporal convolution respectively to capture complicated mixtures of global and local temporal information.

Global convolution temporal component uses a convolutional structure with multiple $T \times 1$ filters to extract time-invariant patterns of all time steps for each driving time series globally. Each filter of the global temporal convolution module produces a vector with a size of $D \times 1$, where the activation function is a ReLU function. Merged by the vectors, global convolution structure outputs an matrix H^G, which is that a well trained representation of a univariate time series can be mapping with each row of the matrix. Local temporal convolution uses the length of the filters as l, where $l < T$ to map local temporal relationships in each univariate time series to a vector representation. Also, a 1-D max-pooling layer is used for each column of the matrix to capture the most representative features.

Then, an self-attention module inspired by the transformer can learn the similarities among the driving series using these representations from each component. The self-attention module

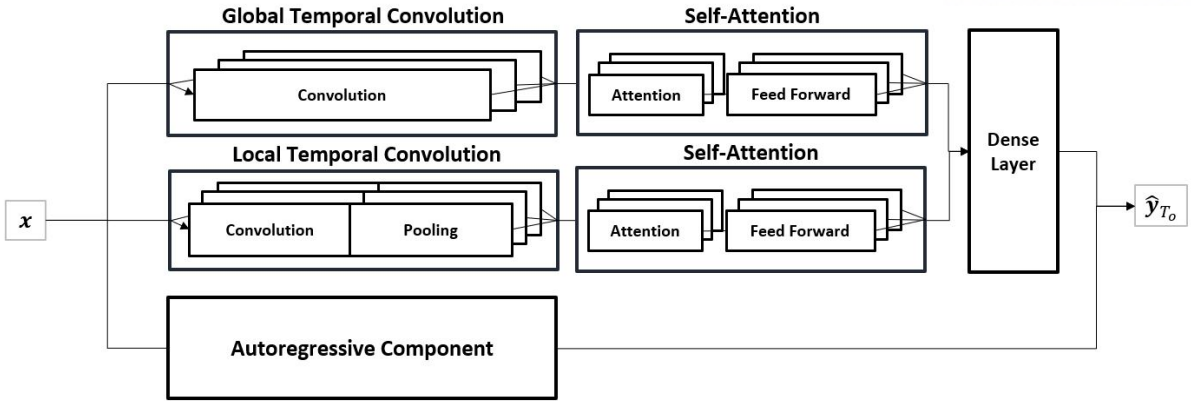

Figure 4: The architecture of Dual Self-Attention Network (DSANet).

is composed of several stack of layers, and two sub-layers which are a self-attention module and a position-wise feed-forward layer is used for each stack layer. This scaled dot product self-attention module which can calculate the similarity between a query and each key to output a set of value is defined as follows,

$$\mathbf{Z}^G = softmax(\frac{\mathbf{Q}^G(\mathbf{K}^G)^\top}{\sqrt{d_k}})\mathbf{V}^G, \tag{14}$$

where $\mathbf{Q}^G, \mathbf{K}^G$, and $\mathbf{V}^G$ are the set of queries, keys, and values obtained by applying projections to the output of global temporal convolution. d_k represents the dimension of the keys. The position-wise feed-forward layer which is contained in two linear layers with a ReLU function is defined as

$$\mathbf{F}^G = ReLU(\mathbf{Z}_O^G W_1 + b_1)W_2 + b_2, \tag{15}$$

where $\mathbf{Z}_O^G$ is the final representation of the self-attention modules. In addition, it has a residual connection followed by layer normalization [46] to make training easier and to improve generalization. To improve the robustness, an Autoregressive(AR) linear model is combined in a parallel manner because the scale of the output in non-linearity methods is not sensitive to the scale of the input.

Finally, the prediction of DSANet is obtained by combining a feed-forward layer to sum among two self-attention modules and the AR prediction.

2.6 L1 Trend Filtering

If a univariate time series y_t, $t = 1, ..., n$ it consists of trend x_t that changes slowly and a random component z_t which changes rapidly. The goal of L1 trend filtering is to estimate the trend component x_t or, equivalently, estimate the random component $z_t = y_t - x_t$. This method can adjust x_t to be smooth and can adjust z_t, referred to as the residual, to be small. The main principle is that the trend filtering optimize trend estimates that are piecewise linear. This method allows the selection of the trend estimate as the minimizer of the objective function as follow,

$$(1/2)\sum_{t=1}^{n}(y_t - x_t)^2 + \lambda\sum_{t=2}^{n-1}|x_{t-1} - 2x_t + x_{t+1}|\,, \tag{16}$$

which can be written in matrix form as

$$(1/2)\,\|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\,\|D\mathbf{x}\|_1\,, \tag{17}$$

where $\|\mathbf{u}\|_1 = \sum_i |u_i|$ denotes the l_1 norm of the vector $\mathbf{u}$ and $D \in \mathbb{R}^{(n-2)\times n}$ is second-order difference matrix.

$$D = \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix} \tag{18}$$

In addition, λ is a non-negative parameter used to handle between the extent of the residual and the smoothness of x. Fig. 5 shows after the trend filtering that the fluctuating behavior of the original time series graph is stabilized and the trend filtering simply follows the general trend between two adjacent knot points. This means that time series models can detect some important signals more easily and that filtering simplifies the prediction of noisy time series. The important signal represents a change point (knot) when the time series trend changes. Many experiments in this paper show the effect of the trend filtering for three deep temporal neural network models.

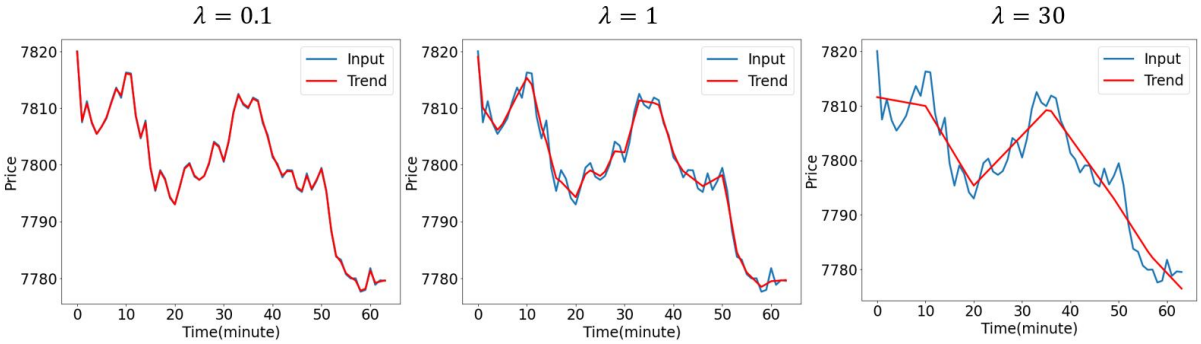

Figure 5: Stock prices before and after the L1 Trend Filtering. Blue line is original value and red line is the optimal trend of original time series. The greater the value of λ, the smoother is adapted to the trend. This makes the number of knots to be smaller.

To derive a Lagrange dual of the primal problem of minimizing Eqn. 16, we set a new variable z which has constraint $z = Dx$, to obtain formulation as follow,

$$\begin{aligned} &\textit{minimize} \quad (1/2)\,\|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\,\|\mathbf{z}\|_1\,, \\ &\textit{subject to} \quad z = D\mathbf{x}. \end{aligned} \tag{19}$$

According to a dual variable ν with the constraint, the Lagrangian is

$$L(\mathbf{x}, \mathbf{h}, \nu) = (1/2)\,\|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\,\|\mathbf{z}\|_1\,, + \nu^T(D\mathbf{x} - z). \tag{20}$$

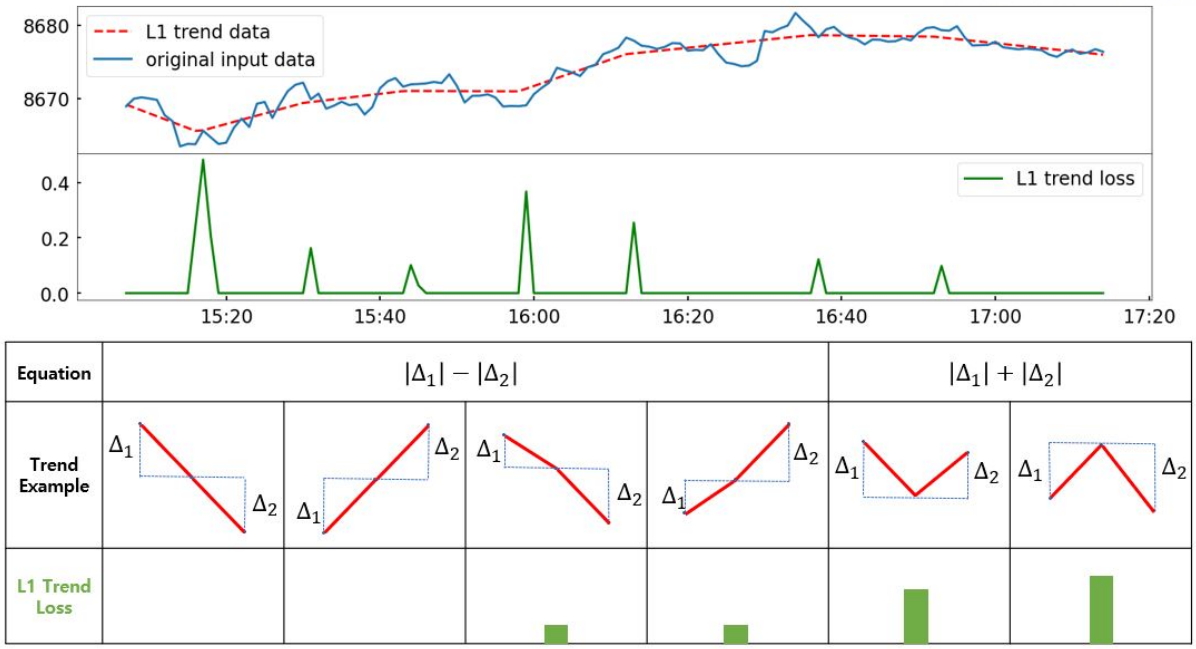

Figure 6: L1 trend loss for various example which have a lot of slopes between the previous and the following points. The greater the difference of slope between the previous and the following points, the greater the loss of the L1 trend (z), which is second term of the smoothness in Eqn. 16. By setting hyperparameter λ, the knot having lower L1 trend loss can be flattened and other point can be remain as knot.

The dual problem is

$$\begin{aligned} \textit{minimize} \quad & g(\nu) = \quad (1/2)\nu^T DD^T\nu - y^T D^T\nu \\ \textit{subject to} \quad & -\lambda\mathbf{1} \leq \nu \leq \lambda\mathbf{1}. \end{aligned} \tag{21}$$

The dual problem Eqn. 21 is a convex and quadratic program (QP). From the solution ν^{lt} of the dual Eqn. 21, L1 trend estimate is calculated as follow,

$$\mathbf{x}^{lt} = \mathbf{y} - D^T\nu^{lt}. \tag{22}$$

III Improved Predictive Model with Trend Filtering Feature

As followed figure 7, a new framework is proposed utilized deep temporal neural networks with a additional trend filtering feature, which is the trend of the target value. Three deep temporal neural networks, which is state-of-the-art in forecasting future time series data, are utilized for comparing between the models with trend filtering feature as an input data and the models without one.

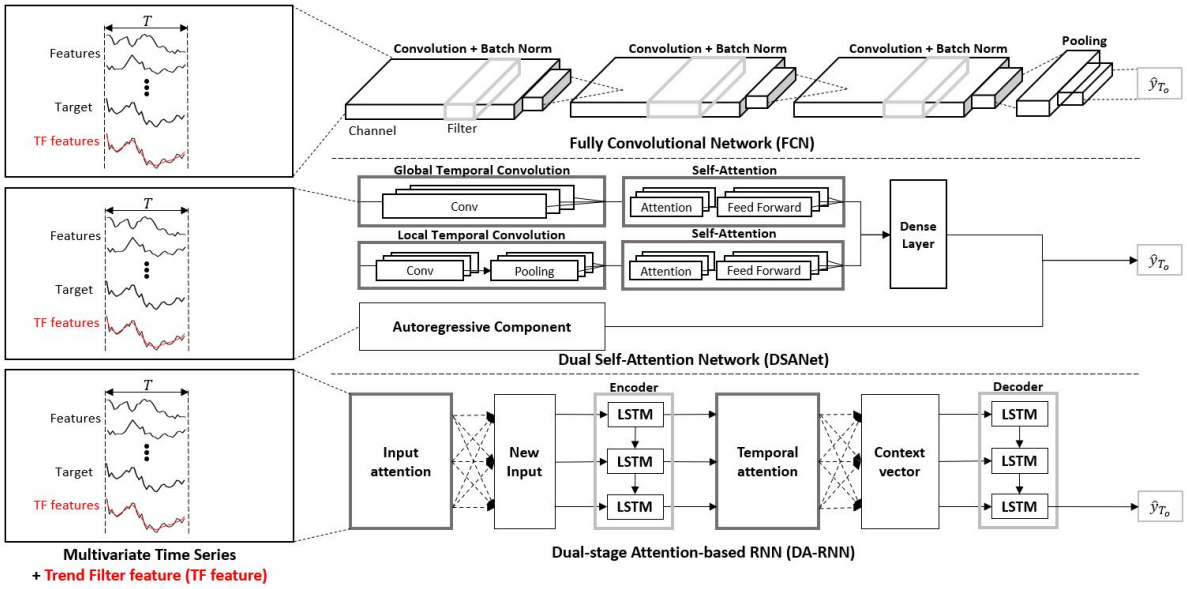

Figure 7: The proposed framework add a feature using trend filter technique. Three deep temporal neural network which are Fully Convolutional Network (FCN), Dual Self-Attention Network (DSNet), and Dual-stage Attention-based RNN (DA-RNN), are used to verify the efficiency of this method. Trend filter feature (TF feature) which is used for input data to predict fucture values is applied by the trend filter technique to original target data. Each predictive values have same time steps for each model, several metrics such as RMSE, MAE, MAPE are used for comparing between with and without trend filter feature.

IV Experimental results

We implement the proposed methods and the baseline models in the PyTorch framework. This sections describes five stock index datasets, after which the parameter settings of the proposed methods and the evaluation metrics are introduced. Lastly, the performance between the proposed methods and original methods is compared for proving the effectiveness of this methods.

4.1 Datasets

To verify the effectiveness of the proposed methods, we utilize five stock market index datasets that represent the different stock exchanges from Bloomberg. These datasets are suitable for multivariate time series forecasting. In order to reflect recent market trends, data from July of 2019 to January of 2020 are used here. Each dataset consists of a stock market index as the target of the prediction and the stock prices of companies as the input. We assume that the function used to compute the index out of the individual stocks is unknown. Thus, the deep neural network models must learn a predictive index function from observations. The frequency of the data is one minute. This satisfies the conditions of the deep neural networks, which require a large number of samples. Moreover, each dataset is divided into two cases: one with L1 trend filtering applied and the other with original inputs. As Table **??**, the information of each datasets is shown.

NASDAQ 100 Stock

The NASDAQ is a separate index of only 100 blue-chip companies listed on the NASDAQ index in the USA. This data covers the term of July 1, 2019 to January 10, 2020, *i.e.*, 193 days in total. In this paper, the first 45,668 data samples is used as the training set and the following 5,075 data samples is used the test set. The growth rate of the NASDAQ 100 Index during the period of test data is 3.26%.

EURO STOXX 50 Index

The EURO STOXX 50 Index represents the 50 leading stocks in 12 Eurozone (*e.g.*, Austria, France, Germany) countries and major sectors. It is calculated by the STOXX company. This data covers the term from July 1, 2019 to January 10, 2020, *i.e.*, 193 days in total. In this paper, the first 65,349 data samples is used as the training set and the following 7,261 data samples is used as the test set. The growth rate of the EURO STOXX 50 Index during the period of test data is -3.47%.

Dow Jones Industrial Average (DJIA)

The DJIA consists of only 30 blue-chip companies listed on the NYSE and NASDAQ indices in the USA. This data covers the term from July 2, 2019 to January 31, 2020, 195 days in total. In

Dataset	Stock Exchange	Size	
		Train	Test
NASDAQ 100 Stock	NASDAQ	45,668	5,075
EURO STOXX 50 Stock	Eurozone	65,349	7,261
Dow Jones Industrial Average	NYSE, NASDAQ	48,740	5,416
FTSE 100 Stock	LSE	61,907	6,879
TSX 60 Stock	TSE	49,404	5,490

Table 1: Information of the datasets.

this paper, the first 48,740 data samples is used as the training set and the following 5,416 data samples as the test set. The growth rate of the DJIA during the period of test data is 1.16% .

Financial Times Stock Exchange (FTSE) 100 Index

The FTSE 100 index represents the stock prices of 100 companies in the order of market capitalization listed on the London Stock Exchange(LSE). The FTSE 100 Index is the leading index of the UK stock market. This data covers the term from July 1, 2019 to January 10, 2020, *i.e.*, 193 days in total. In this paper, the first 61,907 data samples is used as the training set and the following 6,879 data samples is used as the test set. The growth rate of the FTSE 100 Index during the period of test data is -4.26%.

Toronto Stock Exchange (TSX) 60 Index

The TSX 60 Index is a stock market index of 60 large companies in the order of market capitalization listed on the Toronto Stock Exchange (TSE). This data covers the term from July 1, 2019 to January 10, 2020, *i.e.*, 193 days in total. In this paper, the first 49,404 data samples is used as the training set and the following 5,490 data samples is used as the test set. The growth rate of the TSX 60 Index during the period of test data is -0.16%.

4.2 Parameters and Investment Model Settings

To apply the trend filtering feature to the input data, we set the parameter λ to 0.005. In ARIMA, we set the parameters p, d, and q to corresponding values of 1, 0, and 0. In the DA-RNN, there are four parameters, which are the number of input window size T_i, the number of output window size T_o, and each hidden dimension of the encoder m and decoder p. To find optimal parameters, we conducted a grid search. Finally, we find the best performance which are $T_i = 64, T_o = 5$, and $m = p \in \{64, 128\}$. For the parameter T_o, the higher the value of the parameter is, the more often a lagged prediction compared to the ground truth can be detected. Accordingly, this parameter value is fixed. For the other parameters m and p, these parameters are modified proportionally relative to the number of driving time series for each dataset. In DSANet, the

number of multi-head n_{head} is set to 8 and both the inner-layer dimension of Position-wise Feed-Forward Networks and the output of the dimensions are modified proportionally to the number of driving time series for each dataset. In FCN, the parameter filters are 32 in all layers, and the kernel sizes are 7, 5, and 3 in the sequence layer. Since FCN does not perform well in this experiments, only the index value is used as an input. To compare the results between the model with the trend filtering feature and that without it, only the trend filtering feature is added with the index value. To ensure that the prediction is properly done in the desired direction, a simple investment model that buys or sells only one amount for each trade is applied to the output of the regression model. The position is determined by comparing the last predicted values $\hat{y}_{T_o}$ with the actual value y_T. Also, the initial balance is set proportionally to the initial price for each dataset.

4.3 Evaluation Metrics

To evaluate the performance capabilities of various models for forecasting time series, three metrics are used: the root mean squared error (RMSE), the mean absolute error (MAE), and the mean absolute percentage error(MAPE), with the Adam optimizer. RMSE is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_t^i - \hat{y}_t^i)^2}, \tag{23}$$

and MAE is defined as

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_t^i - \hat{y}_t^i\right|, \tag{24}$$

where y_t is the target value at time step t and $\hat{y}_t$ is the predicted value at time step t. MAPE is chosen as an evaluation metric because the proportion of prediction deviation can measure from the ground truth. MAPE is defined as

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_t^i - \hat{y}_t^i}{y_t^i}\right|. \tag{25}$$

In the investment model, the rate of return, used as an evaluation measure, is computed by subtracting the initial balance from the previous balance, adding the value of the stocks an investor has, and dividing these numbers by the initial balance.

$$Rate\ of\ Return(\%) = \frac{B_{Last} - B_{Initial} + Stocks \times P_{close}}{B_{Initial}} \times 100, \tag{26}$$

where B_{Last} and $B_{Initial}$ are correspondingly the last balance and initial balance, and P_{close} is the closing price of the index.

4.4 Lookahead Baseline Method

The Lookahead baseline method is used for performance comparisons of both the prediction accuracy and Rate of Return. As Fig. Fig. 11, the prediction metric is calculated between the

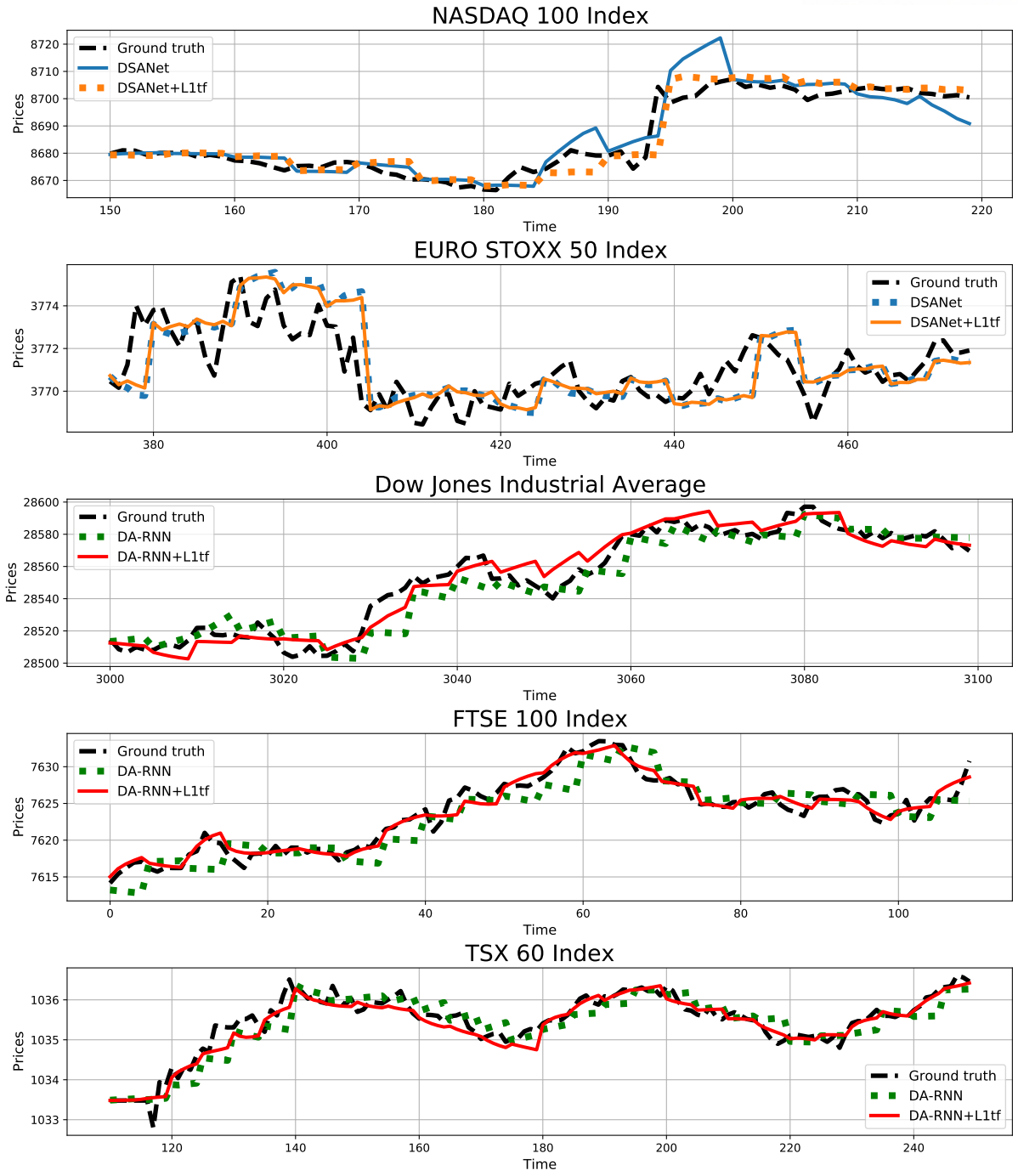

Figure 8: Prediction results for the comparison between the model with the L1 trend filtering and without. Each dataset is located in each row, and each figure shows the prediction difference for two methods of the model.

samples of the shifted value by prediction step and the current value. The rate of return is calculated identically to how it is done in a simple investment model. One difference between the Lookahead baseline method and other models is that the predicted value $\hat{y}_{T_o}$ is not used for determining the current position. The Lookahead baseline method only uses actual prices of test

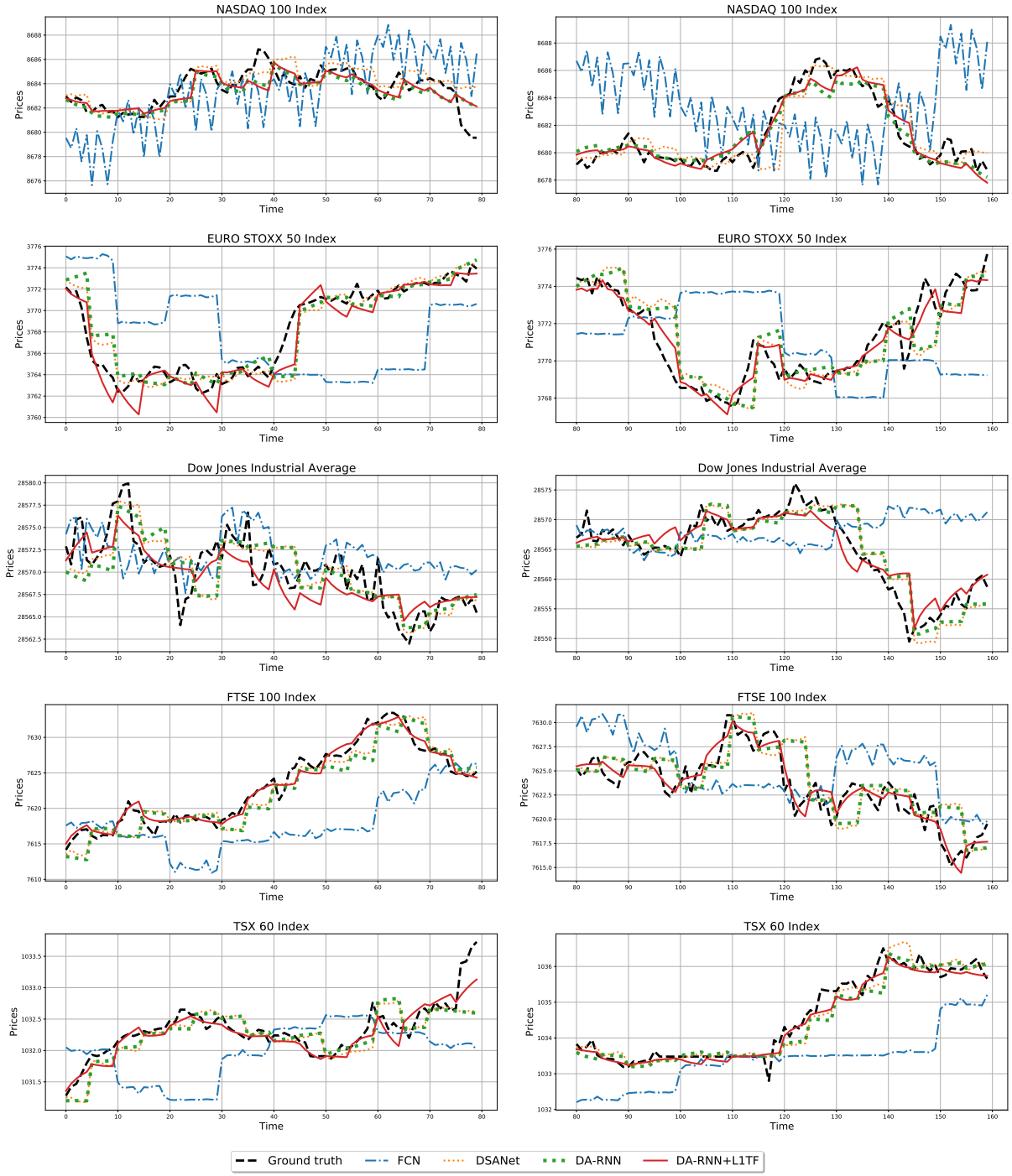

Figure 9: Prediction results of DA-RNN+L1TF compared to other methods. Each dataset is located in each row having two different terms of time by representing two columns, and the prediction comparison from each figure is shown. As shown in the figure, the DA-RNN+L1TF outperforms other methods.

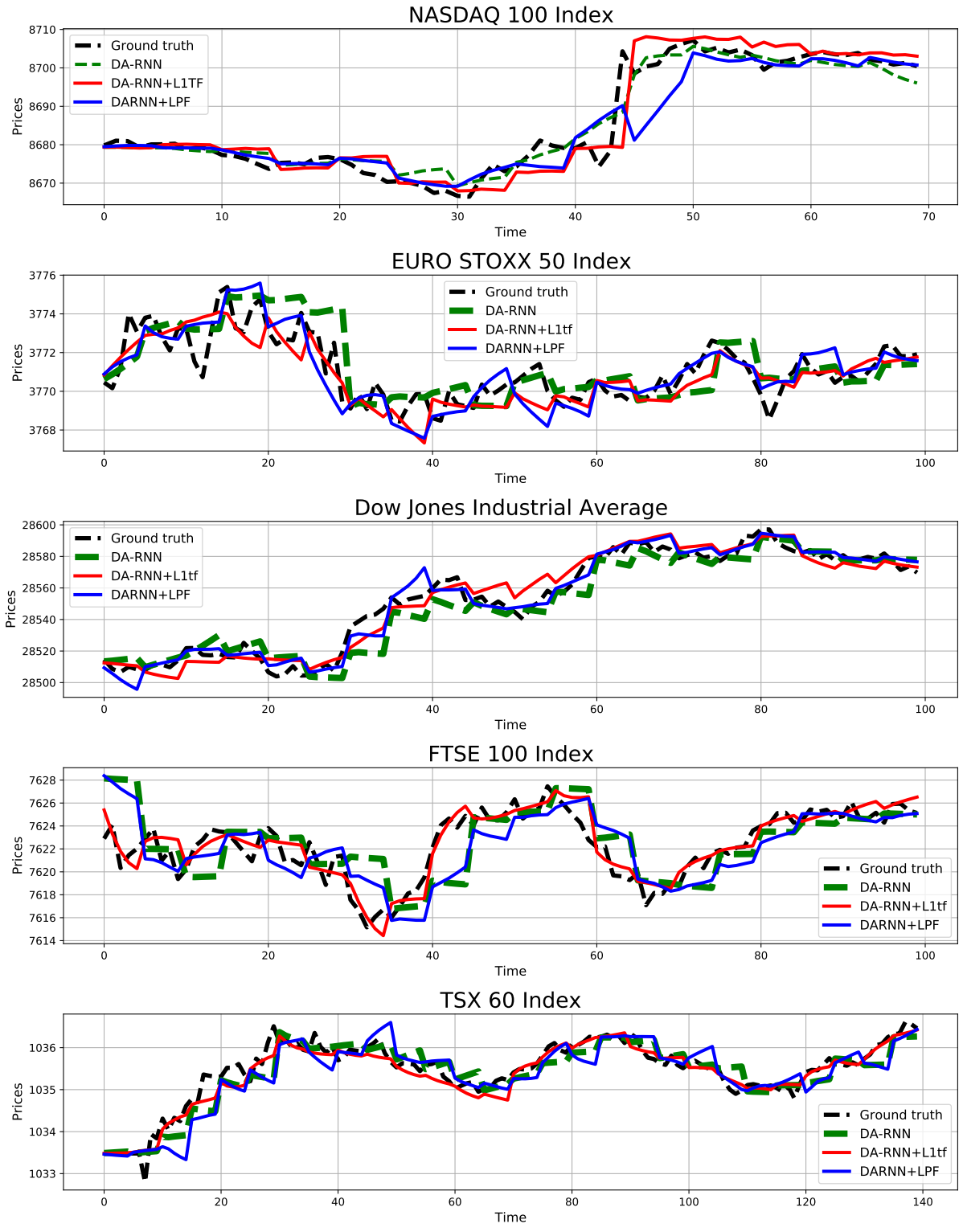

Figure 10: Prediction results of DA-RNN+L1TF compared to DA-RNN+LPT. As shown in the figure, the DA-RNN+L1TF outperforms other methods because DA-RNN+LPT predicts the future values sensitively compared with DA-RNN+L1TF.

Dataset (Growth Rate %)	Model	Metric			
		RMSE	MAE	MAPE	Rate of Return (%)
NASDAQ 100 Index (3.26%)	Lookahead	252.65	10.2805	11.52	0
	ARIMA	8.5656	4.4327	5.02287	1.01
	FCN	9.2603	5.0765	5.7540	7.02
	FCN + L1TF	8.8692	5.2586	5.9606	7.02
	DSANet	5.4383	2.6163	2.9638	6.99
	DSANet + L1TF	5.3484	2.5368	2.874	**7.05**
	DARNN	5.3079	2.6057	2.9514	-3.93
	DARNN+LPT	6.3020	2.5780	2.9190	**7.43**
	DARNN + L1TF	**5.2053**	**2.5146**	2.8486	-0.71
EURO STOXX 50 Index (-3.47%)	Lookahead	85.681	3.5038	9.5152	0
	ARIMA	3.8829	2.1286	5.6934	0.82
	FCN	3.9643	2.2349	5.9765	1.08
	FCN + L1TF	3.4165	1.9012	5.0851	1.04
	DSANet	2.0377	1.1880	3.1777	-1.85
	DSANet + L1TF	2.0049	1.1619	3.1077	-0.18
	DARNN	2.0232	1.1764	3.1469	-0.14
	DARNN+LPT	2.0210	1.1590	3.1010	1.21
	DARNN + L1TF	**1.8392**	**0.9368**	**2.5057**	**7.39**
Dow Jones Industrial Average (1.16%)	Lookahead	787.98	29.0741	10.0784	0
	ARIMA	19.7453	10.4543	3.6458	-1.21
	FCN	20.4051	10.8882	3.7968	9.83
	FCN + L1TF	17.6532	9.7217	3.3899	9.83
	DSANet	12.0051	6.3011	2.1974	9.69
	DSANet + L1TF	12.0281	6.3009	2.1973	12.41
	DARNN	12.6665	6.6695	2.3258	-0.75
	DARNN+LPT	17.9170	6.4610	2.2250	3.24
	DARNN + L1TF	**11.5129**	**5.9758**	**2.0844**	**8.08**
FTSE 100 Index (-4.26%)	Lookahead	176.34	7.1058	9.6254	0
	ARIMA	6.9347	3.8055	5.0598	-0.55
	FCN	6.9883	4.0133	5.3341	-0.43
	FCN + L1TF	6.2695	3.2788	4.3594	-0.30
	DSANet	3.7534	2.2669	3.0142	-0.25
	DSANet + L1TF	3.7444	2.2648	3.0115	0.33
	DARNN	4.244	2.3906	3.1788	-0.24
	DARNN+LPT	5.6033	2.3179	3.0831	**12.780**
	DARNN + L1TF	**2.9276**	**1.3448**	**1.7874**	5.75
TSX 60 Index (0.16%)	Lookahead	27.976	1.0155	9.8054	0
	ARIMA	0.6146	0.3508	3.3636	0.001
	FCN	0.6678	0.3836	3.6775	-0.003
	FCN + L1TF	0.5886	0.3219	3.0861	-0.003
	DSANet	0.4606	0.2210	2.1185	0.65
	DSANet + L1TF	0.4584	0.2195	2.1044	**0.67**
	DARNN	0.4673	0.2256	2.1627	0.09
	DARNN+LPT	**0.3366**	0.1946	1.8647	0.30
	DARNN + L1TF	0.4031	**0.1451**	**1.3917**	0.30

Table 2: Evaluation results of multivariate time series forecasting and Rate of Return. Compared with the original model, one applying trend filtering technique has low values in terms of RMSE, MAE, and MAPE and is similar or higher in terms of Rate of Return. we adopt LPT only to DA-RNN to compare between LPT and other methods.

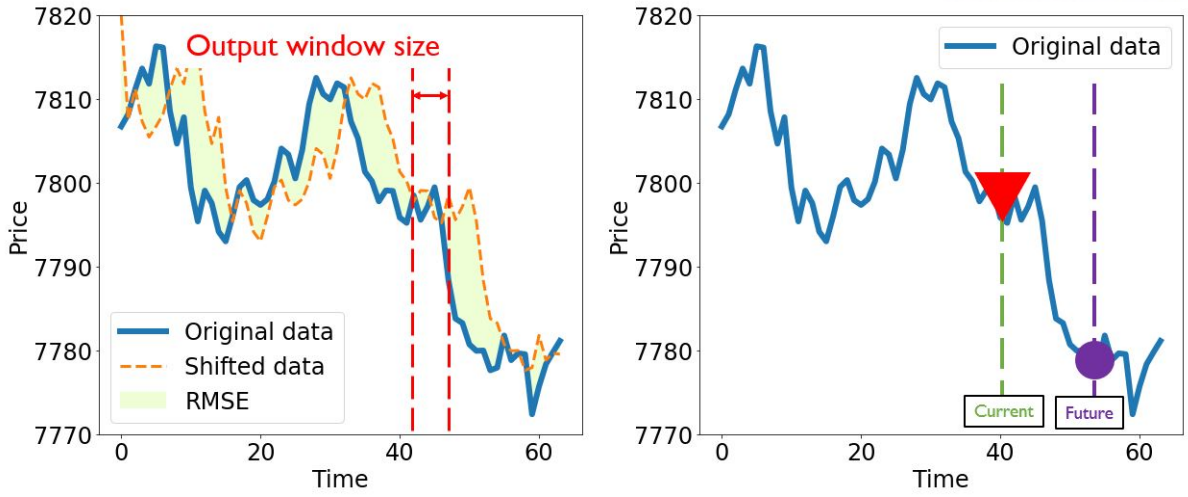

Figure 11: The way for calculating the prediction accuracy (left), and the Rate of Return (right).

data to determine whether an investor buys or sells a stock.

4.5 Prediction Results

We compare the accuracy of Lookahead, ARIMA, FCN, FCN+L1TF (L1 trend filtering), DSANet, and DSANet+L1TF, DA-RNN, DA-RNN+LPT(Low-pass filter), DA-RNN+L1TF models on all datasets. In addition, we conducted several experiments after creating some groups for each model, with one using trend filtering technique, and the other using the original inputs.

Table 2 summarizes the evaluation results of all methods on the test set. In the "model" column in Table 2, "model+L1TF" means that the model utilizes the L1 trend filtering, as described above. For both the prediction and Rate of Return, we observe that the models with trend filtering feautres outperform the models without the trend filtering, indicating that the L1 trend filtering feature is helpful for proper predictions and that it achieves higher performances. In addition, considering that the models with the trend filtering feature obtain better results than the Lookahead baseline method and the growth rate of each dataset, it can be shown that instances of noise in the time series are somewhat distinguishable from informative signals when we use models with the trend filtering feature. In Table 2, DA-RNN+L1FT outperforms the other models in terms of the MAE, and MAPE. In terms of the RMSE on the some datasets, DA-RNN+LPT has better performance. In the FCN case, the larger the step sizes of the input and output are, the more unpredictable the output becomes. Although the DA-RNN and the DSANet show similar prediction results, DA-RNN+L1FT shows much better performance results compared to DSANet+L1TF. To show that DA-RNN+L1TF outperforms the other models, the prediction results for certain methods, including FCN, DSANet, DA-RNN, and DA-RNN+L1TF, are shown in Fig. 10. Only DA-RNN predicts well compared to the other methods 38, and 120 time steps on the FTSE 100 Index and at 48 and, 80 times steps on the TSX 60 Index.

Fig. 8 shows comparisons of the prediction results between the models with the trend

Dataset	Model	w/ L1TF	w/o L1TF	p-value
NASDAQ 100 Index	FCN	8.8692	9.2603	**0.007**
	DSANet	5.3484	5.4383	0.998
	DARNN	5.2053	5.3079	**0.027**
EURO STOXX 50 Index	FCN	3.4165	3.9643	**0.000**
	DSANet	2.0049	2.0377	**0.000**
	DARNN	1.8392	2.0232	**0.000**
Dow Jones Industrial Average	FCN	17.6532	20.4051	**0.000**
	DSANet	12.0281	12.0051	0.466
	DARNN	11.5129	12.6665	**0.000**
FTSE 100 Index	FCN	6.2695	6.9883	**0.000**
	DSANet	3.7444	3.7534	0.380
	DARNN	2.9276	4.244	**0.000**
TSX 60 Index	FCN	0.5886	0.6678	**0.000**
	DSANet	0.4584	0.4606	**0.013**
	DARNN	0.4031	0.4673	**0.000**

Table 3: Results of paired t-test between the models with L1TF and without L1TF. p-values with bold font indicate that the improvements of the models with L1TF over the vanilla models are statistically significant. Out of 15 cases (3 models and 5 datasets), 12 cases are statistically significant.

filtering feature and those without it. In the results with the NASDAQ 100 Index, after the price increases rapidly, DSANet+L1TF predicts the future index value insensitively compared to DSANet without the trend filtering feature. The trend filtering feature is also helpful to predict future time series appropriately in the DJIA, FTSE 100 index, and TSX 60 index results.

In addition, the paired t-test of two methods, with and without the trend filtering, shows the statistical significance of the fact that proposed method achieves better performances than models without filtering. On five datasets and three deep neural network models, twelve results of fifteen cases are statistically significant under significance level 0.05, and the results are shown in Table 3.

V Conclusion

5.1 Summary

In this paper, we proposed a novel method that includes the trend filter feature, such as L1 trend filtering, and Low-pass filter, which is helpful for the task of multivariate time series forecasting, especially for finance data with complex and non-linear dependencies. Furthermore, we applied this method to deep temporal neural networks that can detect certain important signals more easily and filtering simplifies the prediction of noisy time series to address the issue of the difficulty in distinguishing noise from informative signals. Experiments on five index datasets demonstrated that the proposed method outperforms deep neural networks that do not utilize this method for multivariate time series forecasting.

Acknowledgements

I would like to express my gratitude to all people who have been supported me and my research, especially my advisor Jaesik Choi who is a mentor of all my work of the thesis. I would also like to appreciate all members of the Statistical Artificial Intelligence Laboratory. Each of them gives me a lot of helpful advice and their supports. Thank you to my family and friends who always be my side and supports everything without any purpose. Thanks to their support and advice, I can endure everything during Master degree. I would like to express my sincere thank to all of them.

References

[1] Y. Wu, J. M. Hernández-Lobato, and Z. Ghahramani, "Dynamic covariance models for multivariate financial time series," 2013.

[2] P. Chakraborty, M. Marwah, M. Arlitt, and N. Ramakrishnan, "Fine-grained photovoltaic output prediction using a bayesian ensemble," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[3] S. Sahoo and M. K. Jha, "Groundwater-level prediction using multiple linear regression and artificial neural network techniques: a comparative assessment," *Hydrogeology Journal*, vol. 21, no. 8, pp. 1865–1887, 2013.

[4] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 95–104.

[5] S. C. Hillmer and G. C. Tiao, "An arima-model-based approach to seasonal adjustment," *Journal of the American Statistical Association*, vol. 77, no. 377, pp. 63–70, 1982.

[6] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. IEEE, 2014, pp. 106–112.

[7] B. U. Devi, D. Sundar, and P. Alli, "An effective time series analysis for stock trend prediction using arima model for nifty midcap-50," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 1, p. 65, 2013.

[8] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in neural information processing systems*, 1996, pp. 514–520.

[9] Y. Hwang, A. Tong, and J. Choi, "Automatic construction of nonparametric relational regression models for multiple time series," in *International Conference on Machine Learning*, 2016, pp. 3030–3039.

[10] A. Tong and J. Choi, "Discovering latent covariance structures for multiple time series," in *International Conference on Machine Learning*, 2019, pp. 6285–6294.

[11] S. Y. Yang, Q. Qiao, P. A. Beling, W. T. Scherer, and A. A. Kirilenko, "Gaussian process-based algorithmic trading strategy identification," *Quantitative Finance*, vol. 15, no. 10, pp. 1683–1703, 2015.

[12] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[13] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine learning*, vol. 7, no. 2-3, pp. 195–225, 1991.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.

[17] S. Siami-Namini and A. S. Namin, "Forecasting economics and financial time series: Arima vs. lstm," *arXiv preprint arXiv:1803.06386*, 2018.

[18] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[19] K. A. Althelaya, E.-S. M. El-Alfy, and S. Mohammed, "Stock market forecast using multivariate analysis with bidirectional and stacked (lstm, gru)," in *2018 21st Saudi Computer Society National Computer Conference (NCC)*. IEEE, 2018, pp. 1–7.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] J. Li, D. Xiong, Z. Tu, M. Zhu, M. Zhang, and G. Zhou, "Modeling source syntax for neural machine translation," *arXiv preprint arXiv:1705.01020*, 2017.

[22] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[24] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[25] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," 2017.

[26] J. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PloS one*, vol. 15, no. 1, p. e0227222, 2020.

[27] S. Chen and L. Ge, "Exploring the attention mechanism in lstm-based hong kong stock price movement prediction," *Quantitative Finance*, vol. 19, no. 9, pp. 1507–1515, 2019.

[28] Y. Chen, W. Lin, and J. Z. Wang, "A dual-attention-based stock price trend prediction model with dual features," *IEEE Access*, vol. 7, pp. 148 047–148 058, 2019.

[29] L.-C. Cheng, Y.-H. Huang, and M.-E. Wu, "Applied attention-based lstm neural networks in stock prediction," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 4716–4718.

[30] S. Huang, D. Wang, X. Wu, and A. Tang, "Dsanet: Dual self-attention network for multivariate time series forecasting," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2129–2132.

[31] J. Uszkoreit, "Transformer: A novel neural network architecture for language understanding," *Google AI Blog*, vol. 31, 2017.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[33] G. L. Fredendall, "Low-pass filter system," Jun. 14 1949, uS Patent 2,472,798.

[34] S. Adel Sedra and K. C. Smith, "Microelectronic circuits," 1991.

[35] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "$l1$ trend filtering," *SIAM review*, vol. 51, no. 2, pp. 339–360, 2009.

[36] H. Yamada, "A new method for specifying the tuning parameter of 1 trend filtering," *Studies in Nonlinear Dynamics & Econometrics*, vol. 22, no. 4, 2018.

[37] T. K. Kim, "T test as a parametric statistic," *Korean journal of anesthesiology*, vol. 68, no. 6, p. 540, 2015.

[38] H. Akaike, "Fitting autoregressive models for prediction," *Annals of the institute of Statistical Mathematics*, vol. 21, no. 1, pp. 243–247, 1969.

[39] A. Timmermann, "Excess volatility and predictability of stock prices in autoregressive dividend models with learning," *The Review of Economic Studies*, vol. 63, no. 4, pp. 523–557, 1996.

[40] J. S. Hunter, "The exponentially weighted moving average," *Journal of quality technology*, vol. 18, no. 4, pp. 203–210, 1986.

[41] A. Gunasekarage and D. M. Power, "The profitability of moving average trading rules in south asian stock markets," *Emerging Markets Review*, vol. 2, no. 1, pp. 17–33, 2001.

[42] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[43] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[44] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[45] S. Yi, J. Ju, M.-K. Yoon, and J. Choi, "Grouped convolutional neural networks for multivariate time series," *arXiv preprint arXiv:1703.09938*, 2017.

[46] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.